(Non) parliamo di pensiero computazionale[1]

Michael Lodi, Simone Martini, Marco Sbaraglia e Stefano Pio Zingaro
Dipartimento di Informatica - Scienza e Ingegneria, Università di Bologna

1. Introduzione

Espressioni come *pensiero computazionale* ("computational thinking", CT) e *coding* si diffondono sempre più nel mondo della scuola, usate da formatori, libri di testo, documenti ministeriali e leggi nazionali. Questi termini rischiano di generare misconcezioni negli insegnanti (Corradini et al., 2017, 2018) che non hanno ricevuto una formazione rigorosa sui concetti fondamentali dell'informatica, disciplina in cui tali espressioni si collocano.

L'espressione CT ha però origine nella didattica della matematica, probabilmente usata per la prima volta da Seymour Papert nel 1980. Papert sostiene l'uso della programmazione come strumento trasversale, utile agli studenti per rendere concreti concetti astratti e costruire modelli mentali di ciò che stanno imparando. Papert, a partire dal costruttivismo di Piaget, teorizza il *costruzionismo*: il computer, con il suo potere di simulare mondi, mette a disposizione materiali da costruzione diversi e *significativi* (non solo cognitivamente, ma anche emotivamente) per ogni studente. Il CT "alla Papert" evidenzia l'alto valore trasversale dell'informatica, che consente di *eseguire* le astrazioni costruite, ossia di simulare i fenomeni e i concetti da apprendere. Inoltre, il CT è sempre riferito a uno specifico esecutore (in questo si distingue dal *pensiero algoritmico*): "far risolvere" un problema ad un automa esterno, limitato a un ristretto set di istruzioni, che non fa "inferenze umane", è un ottimo esercizio per comprendere a fondo il problema stesso.

D'altra parte, nella società di oggi, permeata di dispositivi e applicazioni digitali e influenzata da algoritmi che governano molteplici suoi aspetti, è indispensabile imparare le basi scientifico-culturali dell'informatica (Lodi et al., 2017). Questo è l'obbiettivo del movimento per il CT a scuola, nato nel 2006 dall'input di Jeannette Wing, che definisce il CT come *problem solving computazionale*: formulare soluzioni in modo che siano eseguibili da un elaboratore di informazioni. Sosteniamo che il CT "alla Wing" sia proprio il "sedimento concettuale dell'informatica" (Lodi et al., 2017).

2. Coding, pensiero computazionale o informatica?

Spesso ci si riferisce alle attività di introduzione giocosa alla programmazione col termine *coding*, che indica solo la fase più meccanica della scrittura del codice (mentre programmare consiste anche in analisi, design, test e debug).
Se si propongono attività di "coding" per sviluppare il pensiero computazionale (cioè per risolvere problemi facendo uso delle idee fondamentali dell'informatica), le attività devono includere l'espressione delle soluzioni in un linguaggio che sia comprensibile ed eseguibile da un automa. Discuteremo due esempi significativi: "pixel art" e "robot sulla scacchiera".
D'altra parte, si può fare *problem solving* senza scomodare l'informatica, così come si può imparare ad orientarsi nello spazio senza robot sulla scacchiera. Allo stesso modo, non ha senso ricondurre tutte le attività che fanno uso del ragionamento logico all'informatica: la logica è un elemento fondamentale del "pensare da informatico" solo se usata affinché un automa elabori informazione.
Piattaforme come CS Unplugged, Code.org e Scratch offrono un ottimo punto di ingresso per insegnare concetti di informatica fin dalla scuola primaria.
Come detto, il *problem solving computazionale* è efficace anche per la didattica di altre discipline. Suggeriremo alcuni esempi nell'ambito della matematica e della fisica: la "geometria della tartaruga", la simulazione di un salto in diverse condizioni di gravità, l'estrazione di grandi quantità di numeri pseudocasuali per sperimentare concetti di probabilità spesso non intuitivi.

3. Conclusioni

Quando parliamo di pensiero computazionale dobbiamo sempre considerare due livelli. Da una parte, il CT non è altro che il sedimento culturale dell'informatica: insegniamo principi di informatica e quello che rimane è pensiero computazionale ("alla Wing"). D'altra parte, non dobbiamo dimenticare il livello "alla Papert": quei principi di informatica non possono essere nozioni "esterne", ma devono trasformarsi in materiali da costruzione significativi nelle mani degli studenti, affinché realizzino concretamente (e quindi comprendano) i concetti anche di altre discipline.

Bibliografia

Corradini, I., Lodi, M., & Nardelli, E. (2017). Conceptions and Misconceptions about Computational Thinking among Italian Primary School Teachers. *Proceedings of the 2017 ACM Conference on International Computing Education Research* (pp. 136–144). New York: ACM.

Corradini, I., Lodi, M., & Nardelli, E. (2018). An Investigation of Italian Primary School Teachers' View on Coding and Programming. *Informatics in Schools. Fundamentals of Computer Science and Software Engineering* (pp. 228–243). Cham: Springer International Publishing.

Lodi, M., Martini, S., & Nardelli, E. (2017). Abbiamo davvero bisogno del pensiero computazionale? *Mondo Digitale, 72*(5), art. 2.

Parole chiave: pensiero computazionale; coding; Papert; Wing; informatica.